Date=07/10/2020
Lecture By=Arkesh Jaiswal
Subject ⇒Javascript Functions

| IN PREVIOUS LECTURE **(QUICK RECAP) Date-06/10/2020** | **In Today's Lecture (Overview)** |
|---|---|
| • JavaScript Arrays<br>• JavaScript Array Methods<br>• Popping and Pushing<br>• Pushing<br>• Shifting Elements<br>• Deleting Elements<br>• Splicing an Array<br>• Slicing an Array<br>• Questions for self Practice / CC For The Day | • JavaScript Functions<br>• JavaScript Function Syntax<br>• Function Return<br>• The () Operator Invokes the Function<br>• Functions Used as Variable Values<br>• Local Variables<br>• Questions For Self Practice // Assignment For the Day<br>• Resources For The Lecture |

# JavaScript Functions

---

A JavaScript function is a block of code designed to perform a particular task.

A JavaScript function is executed when "something" invokes it (calls it).

---

## Example

```
function myFunction(p1, p2) {

  return p1 * p2;   // The function returns the product of p1 and p2

}
```

---

# JavaScript Function Syntax

A JavaScript function is defined with the `function` keyword, followed by a name, followed by parentheses ().

Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

The parentheses may include parameter names separated by commas:

(*parameter1, parameter2, ...*)

The code to be executed, by the function, is placed inside curly brackets: {}

```
function name(parameter1, parameter2, parameter3) {
  // code to be executed
}
```

Function parameters are listed inside the parentheses () in the function definition.

Function arguments are the values received by the function when it is invoked.

Inside the function, the arguments (the parameters) behave as local variables.

A Function is much the same as a Procedure or a Subroutine, in other programming languages.

---

# Function Invocation

The code inside the function will execute when "something" invokes (calls) the function:

- When an event occurs (when a user clicks a button)

- When it is invoked (called) from JavaScript code
- Automatically (self invoked)

# Function Return

When JavaScript reaches a `return` statement, the function will stop executing.

If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.

Functions often compute a return value. The return value is "returned" back to the "caller":

## Example

Calculate the product of two numbers, and return the result:

```
var x = myFunction(4, 3);   // Function is called, return value will end
up in x



function myFunction(a, b) {

  return a * b;           // Function returns the product of a and b

}
```

The result in x will be:

```
12
```

# Why Functions?

You can reuse code: Define the code once, and use it many times.

You can use the same code many times with different arguments, to produce different results.

### Example

Convert Fahrenheit to Celsius:

```
function toCelsius(fahrenheit) {
  return (5/9) * (fahrenheit-32);
}
```

## The () Operator Invokes the Function

Using the example above, `toCelsius` refers to the function object, and `toCelsius()` refers to the function result.

Accessing a function without () will return the function object instead of the function result.

### Example

```
function toCelsius(fahrenheit) {
  return (5/9) * (fahrenheit-32);
}
```

## Functions Used as Variable Values

Functions can be used the same way as you use variables, in all types of formulas, assignments, and calculations.

### Example

Instead of using a variable to store the return value of a function:

```
var x = toCelsius(77);
```

```
var text = "The temperature is " + x + " Celsius";
```

You can use the function directly, as a variable value:

```
var text = "The temperature is " + toCelsius(77) + " Celsius";
```

# Local Variables

Variables declared within a JavaScript function, become LOCAL to the function.

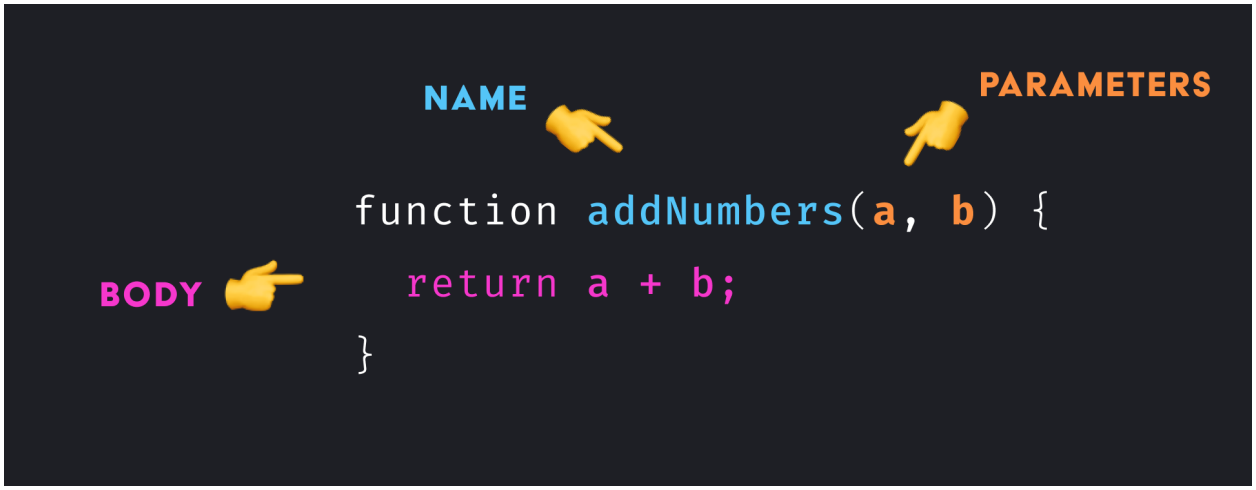
Local variables can only be accessed from within the function.

## Example

```
// code here can NOT use carName



function myFunction() {

  var carName = "Volvo";

  // code here CAN use carName

}

// code here can NOT use carName
```

Since local variables are only recognized inside their functions, variables with the same name can be used in different functions.

Local variables are created when a function starts, and deleted when the function is completed.




# Questions For Self Practice // Assignment For the Day

**https://au-assignment.s3.ap-south-1.amazonaws.com/assignment-fc36e5f2-7d80-4cc1-a90e-751ad38b344a.pdf**

# Resources For The Lecture

https://www.w3schools.com/js/js_function_definition.asp

Youtube Tutorial
https://www.youtube.com/watch?v=N8ap4k_1QEQ